Advanced Computation and I/O Methods for Earth-System Simulations

Nabeeh Jum'ah, Julian M. Kunkel, Anastasiia Novikova, Thomas Ludwig, Thomas Dubos, Sunmin Park, Hisashi Yashiro, Günther Zängl, John Thuburn
(**Contact:** jumah@informatik.uni-hamburg.de; **Management:** j.m.kunkel@reading.ac.uk)

Motivation

- Several groups work on icosahedral-grid based climate/weather models
- Obstacles for Exascale simulations - but also on small scale:
  - Code is very complex and difficult to refactor
  - Climate prediction creates huge data volumes

Limitations of general-purpose programming languages

- Semantics and syntax restrict programmers productivity
- Performance is hardly portable between architectures

Existing Domain-Specific Languages

- May create optimized code for different architectures
- Technical languages with limited relation to scientific domain
- Typically require language-specific paradigm shift for scientists
- Unclear future of the framework/tool

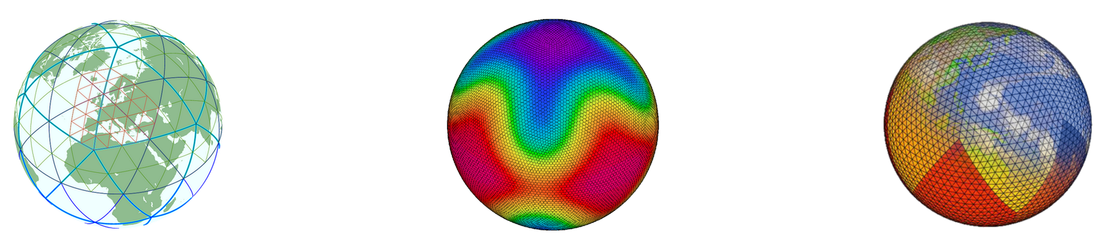
Existing scientific file formats

- Metadata for icosahedral data is not standardized
- Difficult to achieve good performance
- Pre-defined compression schemes achieve suboptimal ratio

Goals

Address issues of icosahedral earth-system models

- Enhance programmability and performance-portability
- Overcome storage limitations
- Provide a common benchmark for icosahedral models



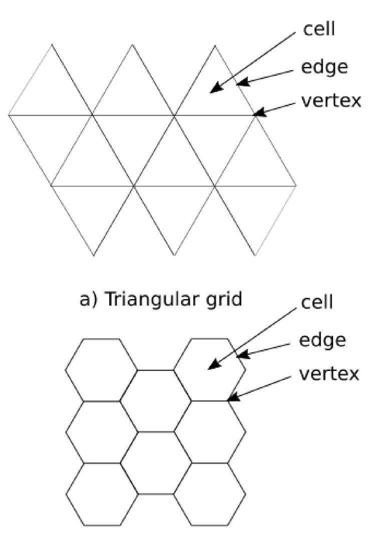
ICON DYNAMICO NICAM

Project Key Facts

- Started March 2016, with three year plan
- Achieved main deliverables:
  - DSL language definition
  - Source-to-source translation tools development
  - SCIL compression library development
  - Icosahedral benchmarks and mini-applications

GGDML Domain-Specific Language

- **GGDML**: the ***General Grid Definition and Manipulation Language***
- Abstracted scientific-domain based constructs for:
  - Data types reflecting "grid" concepts
  - Field declaration
  - Iterators to traverse and update fields
  - Named neighbours in different grids
- Developed in co-design with domain scientists



a) Triangular grid

b) Hexagonal grid

Coding with GGDML

```
foreach c in grid
{
  float df=(f_F[c.east_edge()]-f_F[c.west_edge()])/dx;
  float dg=(f_G[c.north_edge()]-f_G[c.south_edge()])/dy;
  f_HT[c]=df+dg;
}
```

Resulting C code

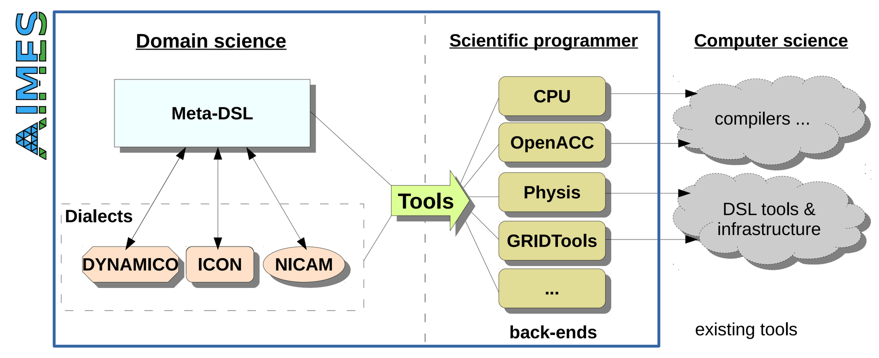
```
    ...handle domain decomposition and halo mangagement
  for (size_t blk_start = (0); ... blocking
    size_t blk_end = ...
#pragma omp parallel for
    for (size_t YD_index = 0; YD_index < local_Y_Cregion; YD_index++) {
#pragma omp simd
      for (size_t XD_index = blk_start; XD_index < blk_end; XD_index++) {
        float df = (f_F[YD_index][XD_index +1] -
                    f_F[YD_index][XD_index]) /dx;
        float dg = (f_G[YD_index +1][XD_index] -
                    f_G[YD_index][XD_index]) /dy;
        f_HT[YD_index][XD_index] = df + dg;
      }
```

- Higher-level code is translated into optimized code, driven by the semantics of the GGDML extensions, and user-provided architecture-specific configurations

Scientific Work Packages: Objectives and Tasks

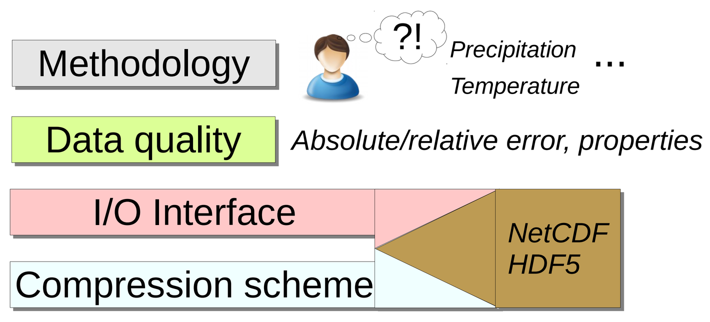
WP 1: Towards higher-level code design

- Foster separation of concerns: Domain scientists, scientific programmer and computer scientists
  - High level of abstraction, reflects domain science concepts
  - Independence of hardware-specific features, e.g. memory-layout
  - Convertible into existing languages and DSLs
- 1.1-1.3 Develop/reformulate key parts of models into DSL-dialects
- 1.4 Design common DSL concepts for icosahedral models
- 1.5 Develop source-to-source translation tool and mappings



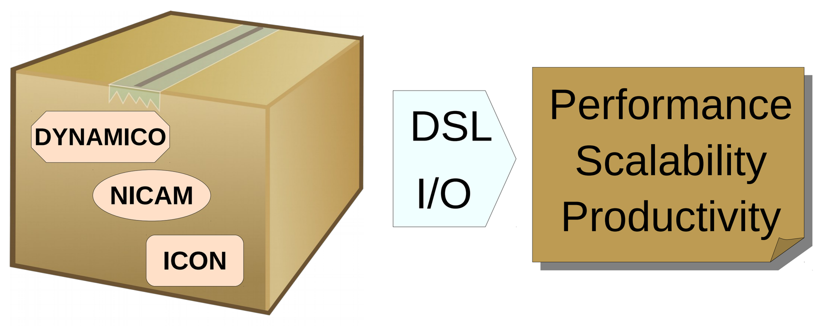
WP 2: Massive I/O

- 2.1 Optimize file formats for icosahedral data
- 2.2 Data reduction concepts
- 2.3 API for user-defined variable accuracy
- 2.4 Identifying required variable accuracy
- 2.5 Lossy compression



WP 3: Evaluation

- 3.1 Selection of representative test cases
- 3.2 Extraction of simple kernels
- 3.3 Common benchmark package/mini-IGCMs[1]
- 3.4 Benefit of the DSL for kernels/mini-IGCMs
- 3.5 Estimating benefit for full-featured models
- 3.6 I/O advances for full models



WP1: Higher-Level Code

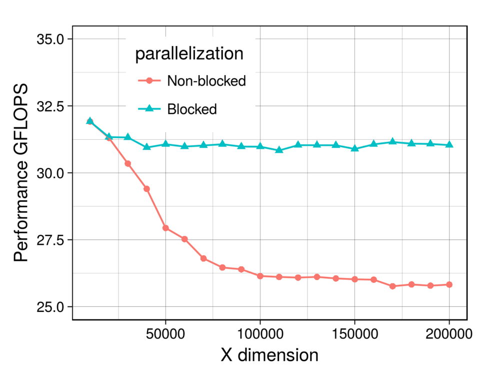
- Milestones:
  - Dialect development: delivered May 2017.
  - The development of the DSL: delivered March 2018.
  - The source-to-source translation tool: continuing.

Architectures and Programming Models

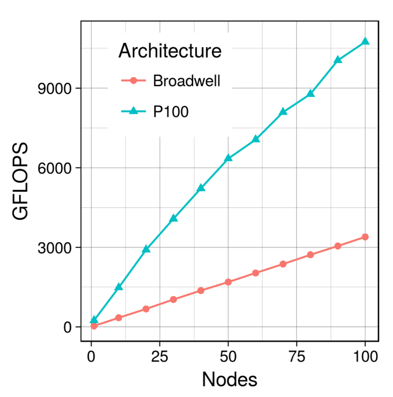
- GGDML code is translated into different targets
  - Multicore processors (with OpenMP)
  - Vector engines (with OpenMP)
  - GPU-accelerated machines (with OpenACC)
  - Multi-node (OpenMP/OpenACC+MPI)

- Recent tool improvements
  - Support for automatic & guided domain decomposition
    - Includes methods for halo exchange
  - Automatic check of dirty halo regions
  - Automatic markup for Likwid instrumentation
  - Function inlining
  - Loop fusion
  - Cache blocking
  - Loop interchange

Experiments



Blocking (Broadwell)



Scaling (Broadwell & P100)

- Blocking improves data reuse with wider grids
- Experiments show the code scales well on CPUs & GPUs

Architecture	Theoretical Memory bandwidth (GB/s)	Before merge: Measured memory throughput (GB/s)	Before merge: GFLOPS	After merge: Measured memory throughput (GB/s)	After merge: GFLOPS
Broadwell	77	62	24	60	31
P100 GPU	500	380	149	389	221
NEC Aurora	1,200	961	322	911	453

Inter-kernel optimization

- Inter-kernel optimization improved application-level performance 35-40% on the different architectures

Architecture	GFLOPS: Scattered	GFLOPS: Short distance	GFLOPS: Contiguous
Broadwell	3	13	25
NEC Aurora	80	161	322

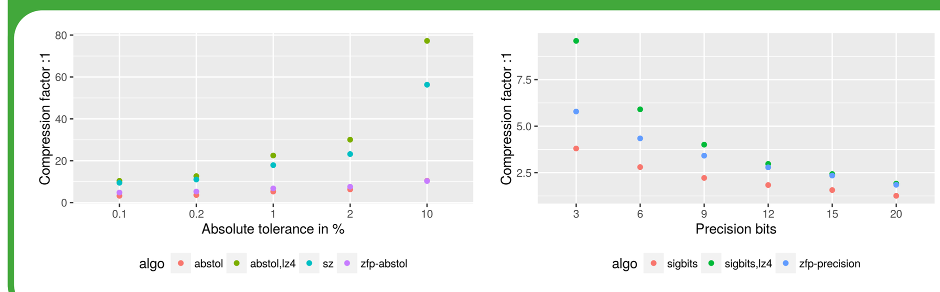
Vectorization and memory layout

- The right memory layout is a key optimization to allow vectorization and efficient use of memory bandwidth

WP2: Compression

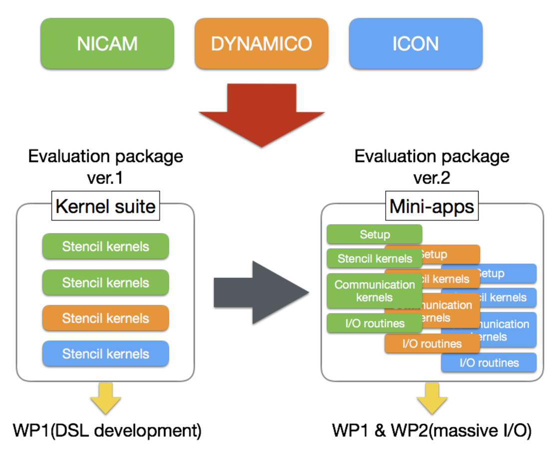
- Development of Scientific Compression Library
  https://github.com/JulianKunkel/scil
- Users define the required accuracy
  - In terms of relative/absolute/precision ...
  - In terms of required performance
  - The library picks a fitting algorithm
- Fill value integration into existing algorithms
- Testing with different models: Isabel, ECHAM, NICAM
- WP 2 status:
  - Extended compression library with new algorithms: delivered 2017
  - Definition of all quantities: delivered 2017
  - Integration into HDF5/NetCDF4: delivered Jan. 2018

Tolerance-Based Results



WP3: Benchmarking

- WP 3 status:
  - IcoAtmosBenchmark v.1 (kernel suites): March 2018.
    https://github.com/aimes-project/IcoAtmosBenchmark_v1
  - IcoAtmosBenchmark v.2 (mini-apps): in progress.



Acknowledgement

This work was supported by the German Research Foundation (DFG) through the Priority Programme 1648 „Software for Exascale Computing" (SPPEXA).

https://wr.informatik.uni-hamburg.de/research/projects/aimes/start